Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de un sistema de simulación de control por voz para una cocina de inducción.

Development of a voice control simulation system for an induction cooker.

Autor

Ana Sierra Moro

Director

José Ramón Beltrán Blázquez

Codirector

David Díaz-Guerra Aparicio

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2020

AGRADECIMIENTOS

Quiero dar las gracias a mi familia, en especial a mis padres y a mi hermano por apoyarme y confiar en mí durante estos años, incluso cuando yo no lo hice en mi misma.

Agradezco a mi tutor José Ramón por darme la oportunidad de hacer este proyecto, con el que tanto he disfrutado; y, sobre todo, a mi codirector David, por resolver todas mis dudas y estar siempre dispuesto a ayudarme en todo lo que he necesitado.

Por último, quiero agradecer a las personas importantes que han estado conmigo durante estos años. A Elena, por llevar juntas desde el primer día de clase, apoyándonos en las largas horas de estudio y por todo lo que hemos pasado. A Borja, gracias por estar siempre a mí lado y por tu apoyo incondicional.

A Miguel, Álvaro, y mis amigas, por hacer todo más fácil y escucharme siempre.

RESUMEN

La tecnología IoT (Internet of Things) supone una revolución en las relaciones entre objetos y usuarios, su crecimiento y desarrollo actual está en auge. Entre sus aportaciones están los asistentes virtuales, diseñados para facilitar la vida a las personas, por esta razón se propone el desarrollo de una interfaz de usuario por voz para el control de una cocina de inducción, con el objetivo de ampliar las posibilidades de su manejo y la comodidad.

Se ha desarrollado una *skill* usando los servicios de Amazon que controla un simulador de una cocina de inducción. El sistema se aloja en la Raspberry Pi que junto con un micrófono y un altavoz simula un altavoz inteligente como, por ejemplo, Amazon Echo. No es un sistema comercial, si no una demostración de las ventajas que podría proporcionar este tipo de servicios y como llevarlas a cabo.

Como conclusión, el funcionamiento de la *skill* es correcto y la comunicación y control se produce con una latencia inapreciable para el usuario.

ABSTRACT

IoT (Internet of Things) technology is a revolution on the relationship between devices and users, currently its growth and development is on the rise. Among its different contributions there are the virtual assistants, designed to make people's lives easier, for this reason it is proposed to develop a voice user interface for the control of an induction cooker, with the aim of extending the possibilities of its operation and user comfort.

A skill has been developed using Amazon services that controls an induction cooker simulator. This system is hosted in a Raspberry PI that together with a microphone and a speaker, simulates an intelligent speaker, such as Amazon Echo. It is not a commercial system, but a demonstration of the clear advantages that this type of service could provide and how to carry it out.

As a conclusion, the operation of the skill is adequate, and the communication and control occurs with an imperceptible latency for the user.

Índice de Contenidos

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este Trabajo de Fin de Grado es lograr el control de una cocina de inducción mediante un asistente virtual comercial controlado por voz, en concreto el sistema: Alexa de Amazon. De esta manera, se pretende poder manejar dicha cocina desde cualquier parte de la casa, simplificando así su uso y aumentando las posibilidades de control al no tener que estar presente en la misma habitación.

Hasta ahora, todas las interfaces de usuario eran interfaces gráficas. Esto es un problema para algunos sectores de la población, como por ejemplo las personas con problemas de movilidad, ceguera, personas mayores que les resulta más complejo usarlas, etc. Las interfaces de voz solucionan estos problemas, la manera más sencilla para entender y expresar las cosas para los seres humanos es decirlo mediante la voz, más rápido, fluido y natural.

Es por eso que este Trabajo de Fin de Grado pretende diseñar un sistema que facilite el uso de una cocina, tanto por la comodidad en general al poder controlarlo desde cualquier zona del hogar mientras se están realizando otras tareas, como para esas personas que antes tenían dificultades para manejarlo.

Este proyecto no plantea un sistema comercial sino una demostración de las ventajas que podría tener este tipo de diseño. Debido a la actual situación de emergencia sanitaria, no ha sido posible trabajar físicamente con una placa de inducción, por lo que se ha optado por desarrollar un simulador *software* cuyo control fuera lo más parecido posible al de los drivers del sistema de *debug* y desarrollo de la cocina que se pensaba utilizar.

El proyecto se realiza en el marco de colaboración que desde el Departamento de Ingeniería Electrónica y Comunicaciones de la Escuela de Ingeniería y Arquitectura se viene llevando a cabo con la empresa BSH.

1.2 Motivación del proyecto

En los últimos años ha habido un crecimiento de la tecnología IoT (*Internet of Things*) que supone una revolución en las relaciones entre los objetos y las personas a través de la red. Es sin duda una de las tecnologías que más van a evolucionar estos próximos años y que cambiará la manera de vivir, trabajar, entretenerse y de interactuar con el mundo.

Su ámbito de aplicación no es exclusivo de los entornos empresariales si no que uno de los campos en los que se está desarrollando es en cómo mejorar la vida cotidiana a través de esta tecnología, y es este el campo de estudio en el que se enmarca este Trabajo de Fin de Grado.

La automatización del hogar, también conocido como domótica, fomenta la accesibilidad ya que facilita el manejo de los elementos del hogar a personas con discapacidades y aporta un mayor confort al hacer que las interacciones sean más rápidas y sencillas.

El motivo principal que ha llevado a la idea de este Trabajo de Fin de Grado es avanzar hacia un futuro en el que las interfaces usuario-voz predominen a la hora de controlar todas las

aplicaciones y elementos de la vida, aumentando así las posibilidades de una gran parte de la población de poder hacer uso de ellas.

1.3 Visión General

En este proyecto se ha propuesto el desarrollo de una interfaz de usuario-voz para el control de una cocina de inducción mediante la plataforma de Amazon Alexa. En lugar de usar un Amazon Echo, que es un altavoz inteligente fabricado por Amazon, el hardware está constituido por una Raspberry Pi 3B+, un *array* de micrófonos USB y un altavoz. De esta manera, la aplicación está instalada en la Raspberry Pi, que es donde se encuentra el *backend* de la *skill*. El *array* de micrófonos es de forma circular y permite realizar una identificación sencilla de la dirección del hablante lo que proporciona una mayor calidad al capturar la voz desde distintos ángulos.

La finalidad de este proyecto no es que la interfaz usuario-voz reemplace a la interfaz física de una cocina, si no que ambas coexistan aumentando así la comodidad y prestaciones de control ofrecidas a los usuarios. Es por esto que en el simulador virtual de la cocina de inducción desarrollado para representar cómo funcionaría en la vida real, el control puede seguir haciéndose manualmente y ambos procesos están conectados. La *skill* desarrollada conoce en todo momento el estado de la cocina gracias a un servidor alojado en la propia Raspberry Pi.

Es importante destacar que no se trata de un sistema comercial que pueda instalarse en todas las cocinas sino de un demostrador que permita mostrar las ventajas y posibilidades de integrar este tipo de interfaces en una cocina. Por tanto, algunas de las soluciones planteadas podrían no escalar correctamente a un sistema comercial.

La comunicación entre el *backend* de la *skill* y el servidor donde se registran los datos se realiza mediante D-Bus, que es un sistema de comunicación entre procesos. Este es el mismo protocolo en el que se basan los drivers del sistema de *debug* de la cocina de inducción con la que se pensaba trabajar, por lo que en un principio bastaría con adaptar las direcciones de los objetos D-Bus para poder sustituir el simulador desarrollado por el control de una cocina real.

Actualmente existen varias plataformas de reconocimiento de voz, entre las que se encuentran la de Amazon (Alexa),Google (OK Google) y Apple (Siri), que permiten utilizarlas con varios dispositivos. Por ahora, Alexa lidera el mercado internacional con casi el 40 % de las ventas de asistentes virtuales [1]. Además, para el control inteligente del hogar Amazon funciona con una gama más amplia de productos.

1.4 Funcionalidad de la cocina

La cocina del proyecto consta de 3 fogones como la mayoría de las vitrocerámicas. Tanto para el simulador gráfico como para la *skill* de Alexa, los posibles controles son:

- Encender cada fuego poniendo a un nivel seleccionado
- Subir y bajar los fuegos de su respectivo nivel
- Apagar cada fuego individualmente
- Apagar todos los fuegos a la vez

1.5 ¿Qué es un asistente virtual?

Es un agente de software que facilita a los usuarios la realización de tareas al usar la mínima interacción hombre – máquina y hacer que este intercambio de información sea más rápido al usar la voz. El usuario hace una petición expresándola en voz alta, y el asistente le responde de igual manera. Los servicios que ofrece pueden ser de todo tipo

- Búsqueda de datos por internet
- Automatización de tareas
- Social media

1.6 Estructura del proyecto

Una vez explicado el objetivo del Trabajo de Fin de Grado y un resumen de cómo se va a llevar a cabo, en los siguientes capítulos se detallan los métodos usados y el procedimiento que se ha seguido.

En el capítulo 2 de *Métodos y Materiales* se presenta la información y las tecnologías usadas para la realización del proyecto, así como los materiales que forman la parte hardware. Tras esto, en el capítulo 3, *Desarrollo del Proyecto* se explica cómo se ha desarrollado el trabajo, los pasos que se han seguido para crear la *skill* y el resto de sistemas. También se exponen los problemas que han surgido y como se han solucionado, y por último el resultado del trabajo.

Finalmente, en el capítulo 4 sobre *Conclusión y Líneas Futuras*, se explica la conclusión del proyecto y el cumplimiento de los objetivos, y las implementaciones y mejoras que podría tener el sistema en proyectos futuros, orientados a un producto comercial.

2. MÉTODOS Y MATERIALES

2.1 Amazon Alexa

2.1.1 Historia del Reconocimiento Automático del Habla.

Uno de los aspectos que ha cobrado más importancia estos últimos años es la comunicación entre el usuario y los equipos a través del reconocimiento automático del habla (RAH), una de las razones es la velocidad a la que se produce la interacción: las personas son capaces de decir más de 200 palabras por minuto, sin embargo la media de palabras escritas por teclado es de 60 [2].

Estos sistemas comenzaron a desarrollarse en los años 50 con un primer sistema que reconocía 10 dígitos de habla inglesa, si se pronunciaban de forma aislada y por el mismo locutor [3]. A principios de 1960 IBM desarrollo “Shoebox”, un precursor de los sistemas de reconocimiento de voz de los que se dispone actualmente que era capaz de entender números del 0 al 9 y hacer las operaciones suma, resta y total, que se le indicaban por comandos de voz [4].

En la década de los años 70 ya se había conseguido el reconocimiento de las palabras de forma aislada, por lo que el objetivo se trasladó a conseguir mantener un dialogo continuado. El proyecto ARPA-SUR fue uno de los mayores trabajos de investigación en RAH, no logró alcanzar sus objetivos pero sirvió para conocer los mecanismos de producción del habla y los verdaderos problemas que conllevaban las técnicas usadas. Por eso, en los años 80, se introdujeron nuevos métodos para desarrollar estos sistemas, entre ellos, las redes neuronales y sistemas capaces de extraer conocimiento de forma inductiva a partir de muestras [3].

En febrero de 2010 se publicó el asistente personal para iOS, Siri, que actualmente es el asistente personal de Apple y está disponible en todas sus plataformas. Un año más tarde del lanzamiento de Siri, en junio de 2011, Google anunció que había incluido la posibilidad de búsqueda por voz. A estas interfaces de voz le siguieron en agosto de 2014 el asistente virtual de Microsoft conocido como Cortana, y a finales de 2014, Amazon anunció Alexa [5].

Actualmente, en altavoces inteligentes, Amazon Alexa lidera el mercado internacional teniendo el 36,6% de ventas. Sin embargo, son muchas las mejoras que quedan por hacer en las interfaces de voz y en estos asistentes [6].

Algunos estudios establecen que la exactitud del reconocimiento de voz varía si lo dice un hombre o una mujer, también varía en función del acento de cada persona, el ruido de la habitación etc. La investigación se centra en descubrir los errores, mejorar el reconocimiento de la voz, la integración de niveles de conocimiento y la modelización acústica.

2.1.2 Reconocimiento de voz Alexa

Alexa es el asistente virtual desarrollado por Amazon, cuyo nombre pretende hacer referencia a la Biblioteca de Alejandría del antiguo Egipto. Además, es necesario que el nombre de invocación sea corto, para que los usuarios lo recuerden con facilidad y de pronunciación sencilla. La ‘x’ en el centro de su nombre le aporta sonoridad lo que hace que al pronunciarlo sea más fácil de reconocer para el asistente virtual.

Está ubicado en la nube de Amazon y disponible en todos sus dispositivos como Amazon Echo, Echo Plus, etc. También se puede instalar Alexa en otros dispositivos como la Raspberry Pi, registrando estos dispositivos como productos en Amazon Voice Service (AVS), así Amazon consigue extender su campo de aplicación al permitir que desarrolladores ajenos a su compañía diseñen nuevas *skills* [7].

Campos principales de aplicación:

- Control del Hogar digitalmente
- Alarmas y temporizadores
- Llamadas y mensajes en manos libre
- Consultas y búsquedas por voz

Figura 1: Control del hogar digitalmente

Figura 2: Alarmas y temporizadores

Figura 3: Llamadas y mensajes en manos libres

Figura 4: Consultas y búsquedas por voz

2.1.3 Amazon Voice Service (AVS)

Amazon Voice Service [7] es un servicio en la nube que permite a los fabricantes de dispositivos integrar las capacidades de voz de Alexa en sus propios productos conectados. De esta manera, da acceso a estos productos a un número creciente de funcionalidades, incluyendo todas las *"Alexa Skills"*. AVS proporciona un servicio de reconocimiento de voz basado en la nube (*Cloud-based Automatic Speech Recognition*) (ASR) y un lenguaje natural comprensivo (NLU).

2.2 Ngrok

Ngrok [8] es un software multiplataforma de túnel y proxy que proporciona un puerto para un servicio de la red, generalmente un servidor web. El puerto está conectado a la nube de ngrok que acepta el tráfico de direcciones públicas y reenvía dicho tráfico a través del proceso que está corriendo en la máquina y finalmente a la dirección local especificada.

Es una manera de establecer un túnel seguro desde un "*Endpoint*" público a un servicio red que se ejecuta localmente.

Se usa para:

- Probar aplicaciones conectadas a un backend que se ejecuta localmente
- Ejecutar servicios personales en la nube desde el hogar
- Obtener direcciones estables para los dispositivos (versión no gratuita)

La principal ventaja de usar este software es que no se necesita configurar nada en el *Firewall* ni en el *router*. Además, permite inspeccionar el tráfico de solicitud y respuesta HTTP a través del túnel creado.

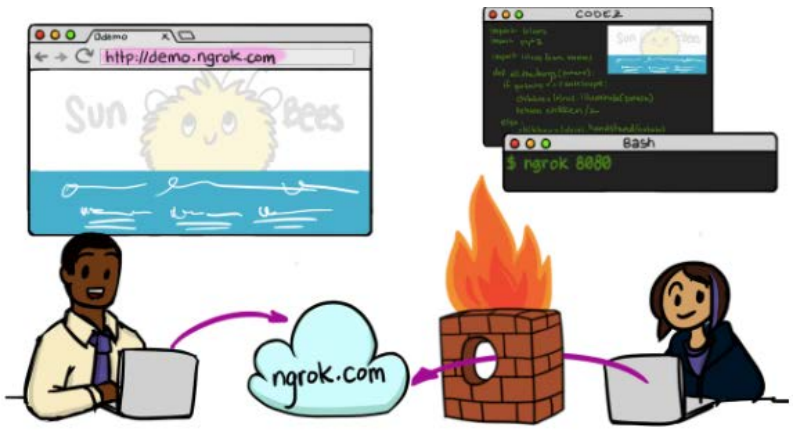

Figura 5: Comunicación a través de ngrok. Imagen sacada de la página oficial de ngrok [8].

2.3 D-BUS

Es un sistema de comunicación entre procesos (IPC) que proporciona una manera sencilla para que las aplicaciones se comuniquen una con otra mediante un sistema de mensajes bus [9].

Fue desarrollado como parte del proyecto de freedesktop.org con el objetivo de normalizar los servicios proporcionados por los entornos Linux y reemplazar los objetos CORBA y DCOP de GNOME Y KDE respectivamente y suplir así las necesidades de un sistema moderno.

D-Bus proporciona un *daemon* de sistema para algunos eventos concretos y un *daemon* de sesión que es el que se usa en general para IPC en las aplicaciones. Consta de una biblioteca, libdbus, que permite a dos aplicaciones conectarse e intercambiar mensajes.

2.3.1 Usos

El objetivo de D-Bus [10] era ser un IPC completo con todas las funcionalidades, por eso tiene varios usos. Permite a un proceso compartir información y datos con otro, también facilita enviar señales o eventos a través del sistema y, por último, implementa un sistema de objeto remoto para que una aplicación pueda hacer peticiones e invocar métodos de un objeto.

La comunicación puede ser entre proceso-proceso o mediante un *deamon* bus formar una topología de bus y permitir así a un proceso comunicarse con una o más aplicaciones a la vez.

2.3.2 Ventajas de usar D-Bus

La unidad básica de D-Bus [10] es un mensaje en lugar de un *stream* de bytes como ocurre en otros mecanismos IPC; es decir, la información se envía en mensajes formados por la cabecera y los datos. El mensaje es binario y al no transmitirse un *stream* de bytes es mucho más sencillo.

Otra ventaja de D-Bus es que, al estar basado en comunicación por bus permite compartir información de un proceso con varias aplicaciones a la vez y no solo entre dos procesos.

Por último, permite la creación de dos buses diferentes: el bus de sistema, que es global y corre a nivel de sistema (de manera que todos los usuarios del sistema pueden comunicarse con este bus si tienen los permisos adecuados) y por otro lado el bus de sesión, que es el que se ha usado en este proyecto y que es creado durante la sesión y corre a nivel de sesión y se utiliza solo por un usuario.

2.4 Interfaz gráfica TKINTER

Las interfaces gráficas son medios visuales que hacen que sea mucho más cómodo la interacción y realización de tareas, en lugar de hacerlo por comandos o texto.

En Python, hay muchas posibilidades para programar una interfaz gráfica de usuario; Tkinter, WxPython, PyQT y PyGTK son algunos ejemplos. Para este proyecto se ha elegido trabajar con Tkinter.

Tkinter [11] [12] es una interfaz gráfica de Python proveniente de la biblioteca TCL/Tk.TCL (Tool Comand Language) es un popular lenguaje de script creado por John Ousterhout con una sintaxis sencilla para aplicaciones embebidas, pruebas, prototipos y desarrollo de interfaces gráficas. Por otro lado, Tk (Tool Kit) es una librería de código abierto multiplataforma que puede ser usado por distintos lenguajes para construir interfaces gráficas.

Aunque Tkinter tiene pocos elementos gráficos (sin listados, árboles, etc.) es relativamente simple y fácil de manejar sobre todo para principiantes. Además, viene pre instalado con Python en casi todas las plataformas y dispone de una gran cantidad de recursos incluidos libros y ejemplos de código, lo que hace que sea una buena elección para interfaces gráficas sencillas de aplicaciones con pocos elementos gráficos.

2.5 Materiales

En lugar de usar un altavoz inteligente desarrollado por Amazon como Amazon Echo, Echo Plus… se ha construido un altavoz inteligente utilizando una Raspberry Pi 3B+, un *array* circular de micrófonos y un altavoz. De esta manera no es necesario publicar la aplicación para tenerla disponible, si no que se registra la Raspberry Pi 3b+ como producto en AVS.

2.5.1 Micrófono: UMA-8 USB mic array

UMA-8 v2 [13] es un micrófono multicanal de alto rendimiento con conformador de haz adaptativo, supresión de ruido, cancelación de eco acústico y de-reverberación, lo que permite reducir los efectos de los sonidos que no son voz (tráfico, ruidos en casa...). Consta de siete micrófonos MEMS, seis de ellos están configurados en una disposición circular para proporcionar alta calidad en la captura de voz desde cualquier ángulo y uno en el centro del dispositivo. De esta forma, se consigue un sistema similar al altavoz de Amazon Echo.

La conexión por puerto USB es imprescindible puesto que la Raspberry Pi tiene solo una salida de audio Jack 3.5mm, por lo que la entrada de audio tiene que ser por USB.

Figura 6: Micrófono

2.5.2 Altavoz

El altavoz que se ha elegido es OK OBS 1060BT-B, es un altavoz de dimensiones reducidas que permite conexión inalámbrica por Bluetooth, cuenta con una entrada auxiliar 3.5 mm Jack, por la que está conectado a la salida de audio de la Raspberry Pi en el proyecto.

2.5.3 Raspberry Pi

La Raspberry Pi es un ordenador de placa reducida (SBC) que consta de una placa base sobre la que se monta un procesador, un chip gráfico y memoria RAM pero no incluye los periféricos.

Hay varios tipos de Raspberry Pi, para este proyecto se ha usado la Raspberry Pi 3 B+ [14] que cuenta con mejor procesador y conectividad que su versión anterior; de esta forma pasa a tener 1,4 GHz, incorpora doble ancho de banda de 2,4 GHz a 5 GHz y un puerto de Ethernet a 300 Mbits/s.

Figura 7: Raspberry Pi 3B+

2.6 Configuración Raspberry Pi

2.6.1 Sistema Operativo

El sistema operativo oficial es Raspberry Pi OS, comúnmente conocido por su nombre anterior Raspbian [15]. Es un software libre basado en una distribución de Linux llamada Debian y optimizado para los productos Raspberry Pi que proporciona el conjunto de programas básicos y utilidades que hace que funcione la Raspberry Pi.

No se trata solo de un OS, si no que incluye unos 35.000 paquetes, software pre-compilado para tener el mejor rendimiento y una experiencia más completa. El sistema se completó en 2012, pero desde entonces está en continuo desarrollo para mejorar la estabilidad y rendimiento todo lo que sea posible . Están disponibles dos versiones para instalar:

- **Raspbian Pixel:** es la versión completa con entorno gráfico, versión de escritorio con menú gráfico más sencilla de usar. (Utilizado en el proyecto)
- **Raspbian Lite:** versión sin el entorno gráfico, en modo consola. Utilizada por usuarios con conocimientos avanzados de programación y proyectos que no requieren interfaz gráfica.

Desde el 2015 la Raspberry Pi Foundation lo ha proporcionado de forma oficial como el sistema operativo primario para la familia de placas SBC.

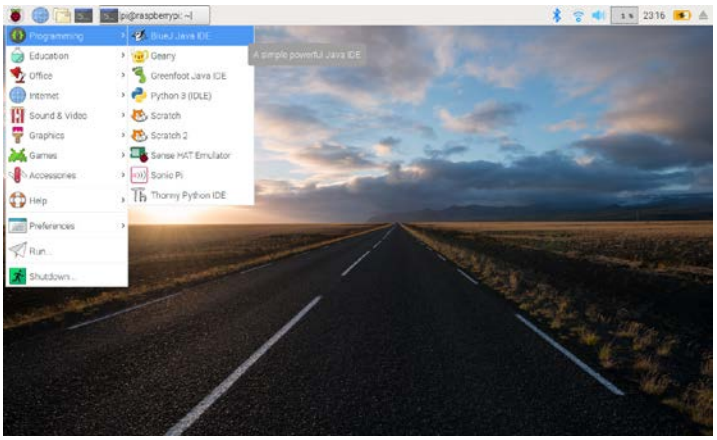

Figura 8: Raspbian Pixel

Figura 9: Raspbian Lite

2.6.2 Instalación del sistema operativo

La distribución oficial se descarga desde el apartado *descargas* de la página oficial raspberrypi.org [16], no debe confundirse con raspbian.org que es la página de la comunidad de innovación de Raspbian. El archivo descargado es una imagen, es decir, un archivo que contiene la estructura y contenidos completos del sistema operativo.

Se necesita una tarjeta micro SD de mínimo 16 Gb para poder instalar el SO y un software para formatear la tarjeta y escribir sobre ella la imagen del sistema operativo descargada. Es necesario formatearla con el formato FAT32, una vez hecho esto se mete la tarjeta en la Raspberry Pi, se conectan todos los periféricos, ratón, teclado, cable HDMI para conectarlo al monitor y se inicia [17].

Es necesario para el proyecto tener conexión a Internet, para mejorar la rapidez en la comunicación se conecta por cable Ethernet. No es necesario tener un monitor adicional para visualizar la pantalla, puede verse desde el propio ordenador mediante un servidor VNC.

2.6.3 VNC server

Es un programa de software libre basado en una estructura cliente – servidor que permite observar las acciones del ordenador servidor y controlarlo remotamente a través de un ordenador cliente. No tiene problemas de compatibilidad en el caso de haber sistemas operativos distintos en el cliente y servidor.

Varios clientes pueden conectarse a un servidor VNC al mismo tiempo.

Figura 10: Esquema de funcionamiento con VNC server

2.6.4 Flask–ask

Flask-ask [18] es un paquete que proporciona una extensión de Flask que permite construir *skills* de una manera más rápida y fácil. Combinada con ngrok, elimina el paso deploy-to-test y permite registrar la invocación de *skills* como un endpoint URL.

Para instalarlo se escribe el comando:

pip install flask-ask

El paquete tiene un paquete de criptografía como una dependencia para la verificación de solicitudes. El paquete de criptografía puede tener requisitos previos adicionales según el sistema operativo [19].

2.6.5 Instalación ngrok y dbus

Instalar ngrok [20] en la Raspberry Pi es muy sencillo. Primero hay que descargar el paquete de su página oficial desde la Raspberry Pi para la versión de Linux, una vez hecho esto, se descomprime la carpeta, y ya está listo para usarlo.

Para instalar d-bus basta con poner los siguientes comandos:

sudo apt-get update

sudo apt-get install d-bus

3. DESARROLLO DEL PROYECTO

3.1 Esquema y Arquitectura del proyecto

En la figura 11, se muestra la arquitectura del proyecto, tanto la conexión y distribución de la parte hardware (MicArray, Altavoz, Raspberry Pi 3 B+) como los protocolos de comunicación que se usan para comunicarse entre los procesos y con Amazon Alexa(D-bus y ngrok respectivamente).

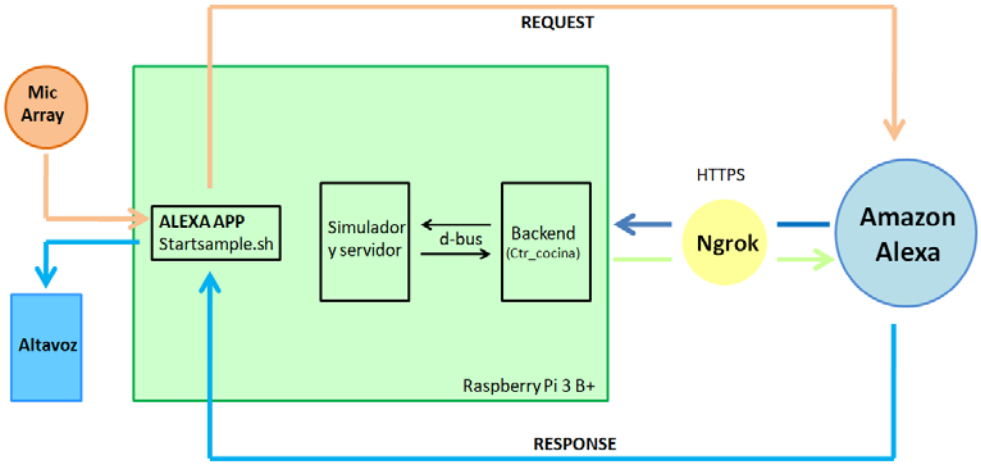

Figura 11: Esquema y Arquitectura del proyecto

3.2 Frontend

El *frontend* de una aplicación hace referencia al lado del cliente, es lo que el cliente ve o a través de lo que el cliente interactúa con la aplicación, se corresponde con las interfaces de usuario ya sean gráficas o de voz.

En este caso se usa una interfaz de voz, el reconocimiento de voz de Alexa. El usuario interactúa con la aplicación mediante un intercambio de frases, primero invoca a la *skill* y después dice lo que quiere hacer. Alexa se encarga de procesar la información y responderle. En el dispositivo del usuario, ya sea un Amazon Echo o una Raspberry Pi como en nuestro caso, se realiza la detección de la palabra clave (Alexa), pero una vez esta es reconocida se envía el audio a los servidores de Amazon donde se realiza realmente el reconocimiento del habla.

Estas interacciones se programan en Amazon Alexa con *Alexa Skills Kit* que permite construir el *frontend*.

3.2.1 Alexa Skills Kit

Es un kit que permite aprovechar el conocimiento de Amazon y la innovación que ha llevado a cabo en el sector de voz para facilitar a los desarrolladores crear nuevas *skills,* gracias a que proporciona un conjunto de herramientas, documentación, ejemplos de código y API que hacen que sea más sencillo y rápido diseñarlas [21].

Hay varios tipos de *skills* que se pueden construir:

- **Custom skill:** maneja cualquier tipo de petición, es el desarrollador el que define las peticiones y las frases que detecta. (Elegida para el este proyecto)
- **Smart Home Skill Api**: este tipo de *skill* está dedicada a aplicaciones para controlar dispositivos inteligentes habilitados para la nube.
- **Video Skill Api:** permite controlar los servicios de video habilitados en la nube.
- **Api de Skill musical:** permite a los usuarios controlar y reproducir el contenido de audio transmitido a través de un dispositivo habilitado para Alexa.

3.2.2 Pasos para construir una skill.

Para diseñar una *skill* se necesita una cuenta de desarrollador de Amazon en "Amazon Developer Services". Una vez que la cuenta esta creada los pasos a seguir son los siguientes:

1. Acceder a la consola de desarrollador desde Amazon Developer Services e iniciar sesión.
2. Seleccionar el botón que aparece en la página con el nombre de ***"Create Skill"***.
3. Elegir el nombre de la *skill*, en este caso se llama *Ctr_cocina_TFG.*
4. Seleccionar el idioma (Español (ES)).
5. Elegir el modelo con el que se va a construir (*Custom*).
6. Elegir donde se va a alojar el *backend* (*Provision your own*, en este caso)
7. Finalmente seleccionar el botón ***"Create Skill"***.

La *skill* se configura en la página Build, dónde se desarrolla y se construye la interacción con el usuario. La figura 12 muestra dicha plataforma, en el lado derecho de la imagen aparecen 4 bloques que corresponden con los pasos que hay que seguir y el orden en el que hacerlo para configurar la skill.

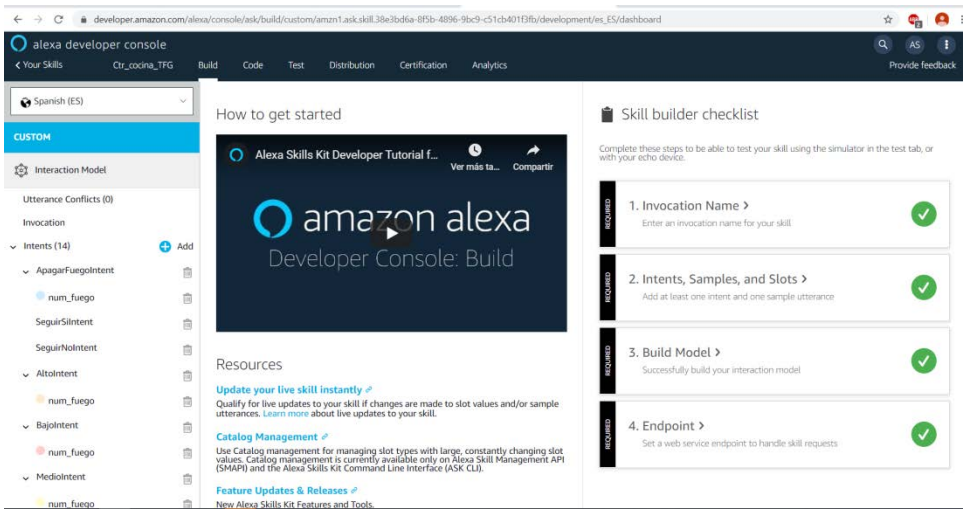

Figura 12: Página 'Build' de la consola de desarrollador.

1. Nombre de invocación

   Es el nombre que tiene que decir el usuario para empezar la interacción con la aplicación. Es importante no confundir el nombre de invocación, con el nombre que le hemos puesto a la *skill* que es el que verán los usuarios si la *skill* se publica a nivel comercial.

Es obligatorio que el nombre de invocación esté formado al menos por dos palabras, y que ninguna de ellas sea las que usa Amazon para lanzar las *skills*, por ejemplo “abre”, “dime”, “Alexa” etc.

El nombre de invocación elegido para la *skill* de este proyecto es *“a cocinar”*, es corto, fácil de pronunciar y de recordar, y tiene relación con el objetivo de la aplicación.

2. *Intents*, *Samples* y *Slots*

   Un *Intent* representa una acción que cumple con una petición del usuario. Los *intents* pueden tener argumentos, conocidos como *slots*, que representan una variable en una petición. Los *slots* pueden ser de varios tipos, están definidos por Amazon, algunos ejemplos son AMAZON.Number, AMAZON.Color, AMAZON.DATE, etc. En esta *skill* solo se ha usado los slots de tipo AMAZON.Number.

   Amazon proporciona *intents* que ya están creados para que el desarrollador pueda incluirlos en su *skill,* su nombre empieza siempre por AMAZON, quedando el nombre completo como AMAZON.NombreIntent. Hay 4 *intents* estándar que se crean automáticamente al crear una *skill*, estos *intents* no pueden tener *slots*, son:

   AMAZON.CancelIntent (detiene la acción de ese momento)
   AMAZON.StopIntent (detiene la *skill*)
   AMAZON.HelpIntent (proporciona ayuda sobre cómo usar la *skill*)
   AMAZON.NavigateHomeIntent

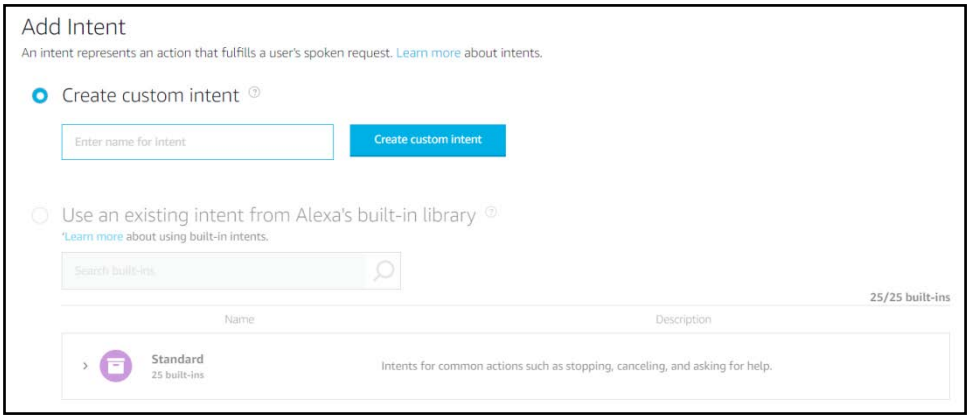

Figura 13: Creación de un nuevo Intent

Cada *intent* se acciona cuando el usuario pronuncia alguna de los *Sample Utterance* asignadas a dicho *intent*. Las *Sample Utterance* son las frases guardadas para cada *intent* y las únicas que reconoce, de forma que cuando el usuario dice alguna de ellas se activa el *intent* correspondiente. Una misma *Sample Utterance* no puede asignarse a más de un *intent*, si no, se produce un conflicto de *Sample Utterance* porque no sabe qué *intent* activar.

Un *intent* puede funcionar con una sola frase asignada, sin embargo, lo que se pretende con los asistentes de voz es facilitar la interacción, por tanto cuantas más frases tenga guardadas más se parece a una conversación real. Además, de esta manera el usuario no tiene que memorizar una única frase con la que interactuar, si no hablar de manera natural y espontánea y que el asistente le entienda.

En la figura 14 se ve un *intent* llamado EncenderFuegoIntent, sirve para detectar cuando el usuario quiere encender uno de los fogones de la cocina y a cuanto quiere ponerlo. Tiene 16 *Sample Utterance* de manera que cada vez que el usuario quiera llamar al *intent* puede hacer uso de cualquiera de ellas. Las variables marcadas en colores representan los slots que tiene, para indicar un slot en las frases basta con escribirlo entre llaves y haberlo creado previamente (Figura 15). Este *intent* consta de 2 slots que son {num_fuego} para indicar el número del fuego que queremos controlar y {num_nivel} para decir a qué nivel queremos encender el fuego. Ambos slots son del tipo AMAZON.Number, que se utiliza para convertir palabras que expresan un número en dígitos.

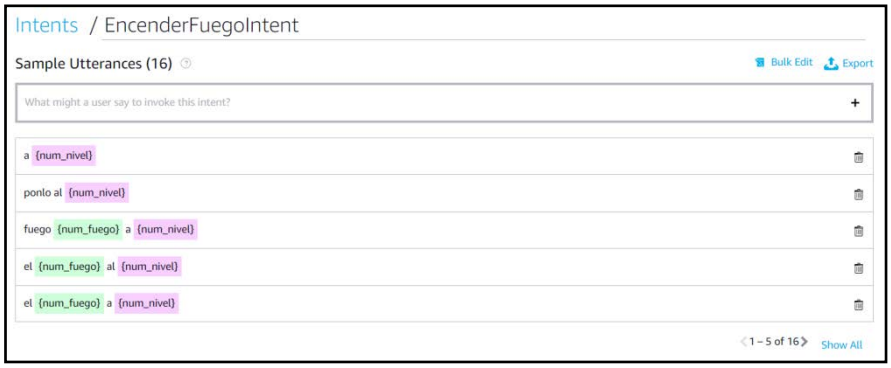

Figura 14: Ejemplo de un *intent* EncenderFuegoIntent

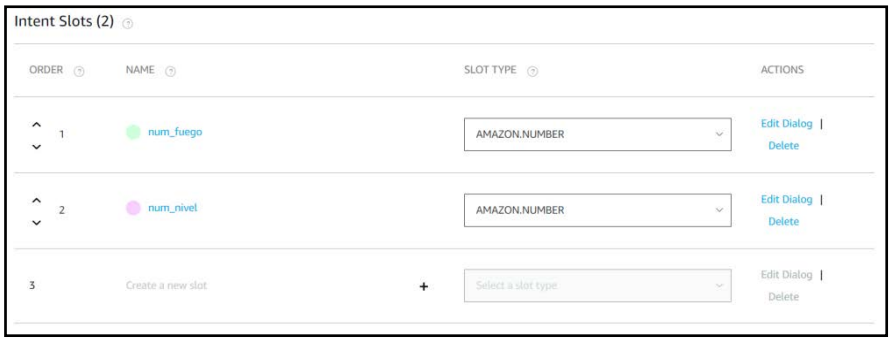

Figura 15: Creación de slots.

Para poder declarar un slot en un *Sample Utterance* es necesario haberlo definido previamente, como se ve en Figura 15. Para añadir un nuevo slot se escribe el nombre con el que se llamará dónde pone "*Create a new slot"* y a continuación, se

selecciona el tipo y se pulsa en el botón "**+**" para agregarlo. Una vez hecho esto, ya está creado el slot y puede usarse como variable en las frases.

A medida que se avanza en la construcción del modelo de interacción se crea un archivo JSON automáticamente que registra toda la información, contiene los *intents*, *slots* y *Sample Utterance.* Es posible editar este archivo y construir el modelo de interacción directamente en JSON en lugar de hacerlo a través de los pasos que proporciona la plataforma. JSON (Java Script Object Notation) es un formato de texto sencillo para el intercambio de datos.

3. BuildMode
   Una vez que se ha desarrollado por completo el modelo de interacción, para hacer efectivos los cambios hay que seleccionar el botón "Build". Si no hay ningún conflicto y está todo completo, se compila la *skill* y ya está disponible para usarla.
   Cada vez que se haga un cambio en el modelo de interacción es necesario hacer este paso.

4. Endpoint
   El *endpoint*, "punto final", es la URL del servicio *backend* que responde a los *intents* y especifica dónde se encuentran los recursos de una aplicación. El usuario cuando interactúa lo hace con el frontend, no ve el *backend* ni el *endpoint*.

   La forma más sencilla de construir una *skill* es alojar el *backend* en AWS Lambda, un servicio que ofrece Amazon Web Services que permite ejecutar el código en la nube sin administrar servidores, y tener el *endpoint* asociado a AWS Lambda. Sin embargo, este no es el caso de este proyecto, en el que el *endpoint* está situado en la Raspberry Pi para poder interactuar directamente con la cocina (o finalmente con el simulador). Esta opción da la posibilidad de una mayor personalización, además de evitar llegar al límite de recursos que proporción AWS de forma gratuita.

3.2.3 Intents de la skill

A continuación se explican todos los intents que se han creado para la skill, sin tener en cuenta los intents básicos que proporciona Amazon.

1. ApagarFuegoIntent: al invocar este *intent* se indica el fuego que se desea apagar. (Figura 20, ver ANEXO I).
2. EncenderFuegoIntent: con este *intent* se selecciona el fuego que se desea encender y el nivel al que ponerlo. (Figura 21, ver ANEXO I).
3. SubirFuegoIntent: el usuario expresa su deseo de subir el fuego a un determinado nivel. (Figura 22, ver ANEXO I).
4. BajarFuegoIntent: el usuario pide bajar el fuego a un determinado nivel. (Figura 23, ver ANEXO I).
5. AltoIntent: es común a la hora de cocinar, usar frases como "ponlo a fuego alto" sin especificar un número concreto, este intent pretende cumplir con esa función y

detectar si el usuario hace ese tipo de petición. Por definición del sistema considera fuego alto el nivel 8. (Figura 24, ver ANEXO I).

6. BajoIntent: se invoca este intent cuando el usuario solicita poner un determinado fuego, a "fuego bajo", se corresponde con el nivel 3. (Figura 25, ver ANEXO I).
7. AltoIntent: al activar este intent, el fuego se pone a un nivel 5, que es el que se corresponde con el concepto de "fuego medio". (Figura 26, ver ANEXO I).
8. MaxIntent: si el usuario previamente ha solicitado poner un fuego, a un nivel mayor de 9, el asistente virtual le indica que esto no es posible, y que solicite un nivel igual o menor de 9, es en ese caso cuando se activa este intent, esperando como utterance un número. (Figura 27, ver ANEXO I).
9. SeguirSi: el usuario desea hacer alguna petición más. (Figura 28, ver ANEXO I).
10. SeguirNo: el usuario no desea seguir con la sesión y se cierra. (Figura 29, ver ANEXO I).
11. ApagarTodoIntent: al activar este intent se apagan todos los fuegos de la cocina. (Figura 30, ver ANEXO I).

3.3 Backend

El *backend* es la parte de la aplicación que el cliente no ve, no forma parte de la interacción directa con el usuario. Es donde se encuentra la capa de datos y la lógica de programación de la *skill*. Es necesario que el *backend* soporte conexión *https* para poder comunicarse con el *frontend* de forma segura, recibiendo y enviado información.

El *backend* de esta *skill* está alojado en el *script* ctr_cocina que se encuentra corriendo en la Raspberry Pi. El *script* ctr_cocina contiene el código de la *skill* para saber qué acciones realizar cuando se invoca cada *intent*. Todos los *intents* desarrollados en el modelo de interacción tienen que ser gestionados por el *backend*, en este caso por *ctr_cocina* mediante la librería Flask-ask. En cada uno de ellos se pedirá la información por medio de peticiones *get* al servidor d-bus cuando corresponda, para conocer el estado de la cocina, y se actualizará con la solicitud del usuario por medio de una petición *set.* D-bus resuelve la comunicación entre servidor y backend.

Para poder comunicar la lógica de programación y el modelo de interacción es necesario asignar la URL del *endpoint.* Dado que el *backend* no está alojado en Amazon Web Services (AWS) si no en la Raspberry Pi se necesita una URL pública, para eso se usa la librería ngrok que proporciona una URL y crea un túnel seguro permitiendo recibir peticiones de direcciones públicas que por medio del proceso que está corriendo en la Raspberry Pi las redirige a la dirección local. Para este proyecto no se ha adquirido una URL estática si no que cada vez que se ejecuta el comando de la figura 16 se obtiene una nueva URL.

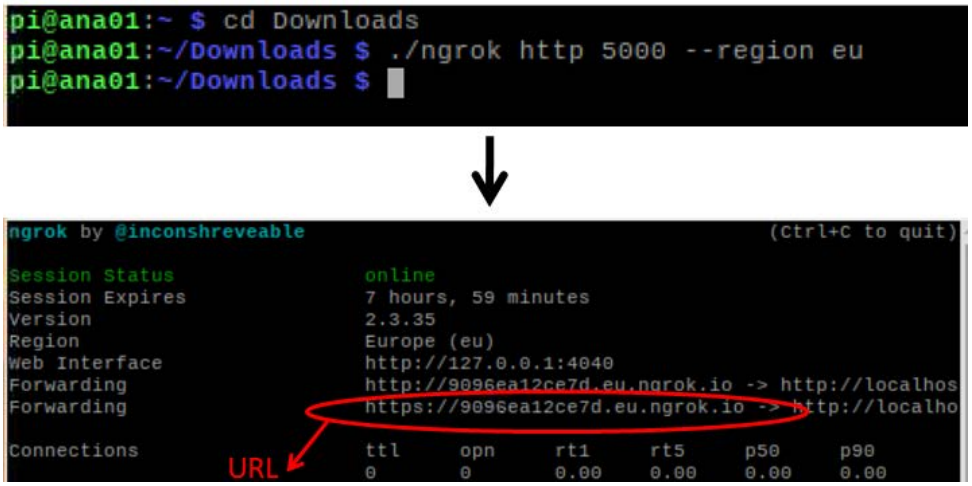

Figura 16: Obtener la URL del endpoint con ngrok.

La configuración del *endpoint* se realiza en el paso 4 de las instrucciones del *frontend* (apartado 3.2.2), la página de la plataforma puede verse en la figura 17.

Existen dos tipos de servicios del *endpoint*, AWS Lambda ARN que es la opción que se elige si se ha usado AWS para alojar en *backend*, o seleccionar HTTPS cuando el *endpoint* se aloja por cuenta del usuario. En el caso de seleccionar HTTPS, se introduce la URL que proporciona ngrok y la opción *"My development endpoint is a sub-domain of a domain that has a wild card certifícate from a certifícate authority"*.

Figura 17: Página de configuración del endpoint de Amazon Alexa.

El *endpoint* recibe peticiones POST cuando el usuario interactúa con la *skill* de Alexa. El cuerpo de la solicitud contiene parámetros que el servicio puede usar y realizar la lógica y generar una respuesta con formato JSON.

3.4 Simulador y Servidor

3.4.1 Simulador

El simulador realizado en este proyecto pretende ser una interfaz gráfica que simule una cocina al no poder disponer de una real. De esta manera se demuestra cómo sería el funcionamiento del proyecto y se observa si la comunicación entre la *skill* y la cocina es correcta.

Para desarrollarlo se ha utilizado Tkinter, la interfaz gráfica estándar de Python dado que no se requieren muchos elementos gráficos. El objetivo ha sido diseñar una placa de inducción común lo más parecida posible a como sería en la realidad, consta de tres fuegos y sus correspondientes botones para controlarla.

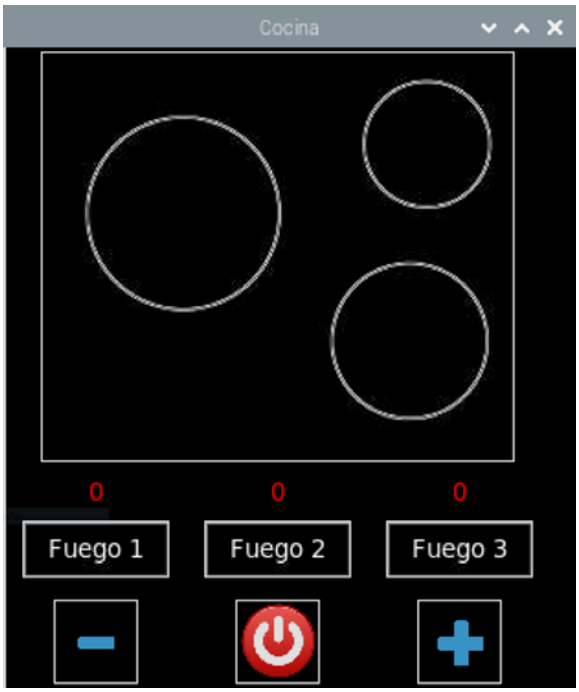

Figura 18: Simulador de la cocina.

En la figura 18 se puede observar el resultado. Tiene seis botones, aquellos cuyo nombre es *"Fuego X"* siendo X un número comprendido en el intervalo [1,3] sirven para seleccionar el fuego correspondiente e indicar que los cambios se harán sobre él.

El botón "–", baja un punto el nivel del fuego que se está modificando, si llega a cero se apaga el fuego, mientras que el botón "+" sube un punto el nivel cada vez que se pulsa sobre él, hasta un máximo de nueve. El botón "off" apaga todos los fuegos a la vez. Cada fuego tiene un dígito en rojo que muestra el nivel en el que se encuentra.

3.4.2 Servidor

El simulador de la cocina tiene que compartir información con el *backend* de la *skill* para conocer el estado en cada momento y reflejarlo así en la GUI del simulador. Esta comunicación se logra por medio de un servidor D-Bus. La *backend* de la *skill* accede a esta información haciendo peticiones *get* al servidor para conocer el estado, y peticiones *set* para actualizarlo con los valores que le solicita el usuario por voz. El simulador y el servidor se encuentran en el mismo *script*, de esta manera el simulador no tiene que hacer peticiones por D-bus y puede acceder y actualizar el estado de las variables del servidor cuando se modifica manualmente. De la misma manera, el servidor cuando recibe una petición de la *skill* modifica las variables de

las que depende la GUI del simulador. En la figura 19 se puede ver un ejemplo de la comunicación cuando se activa el *intent* EncenderFuegoIntent.

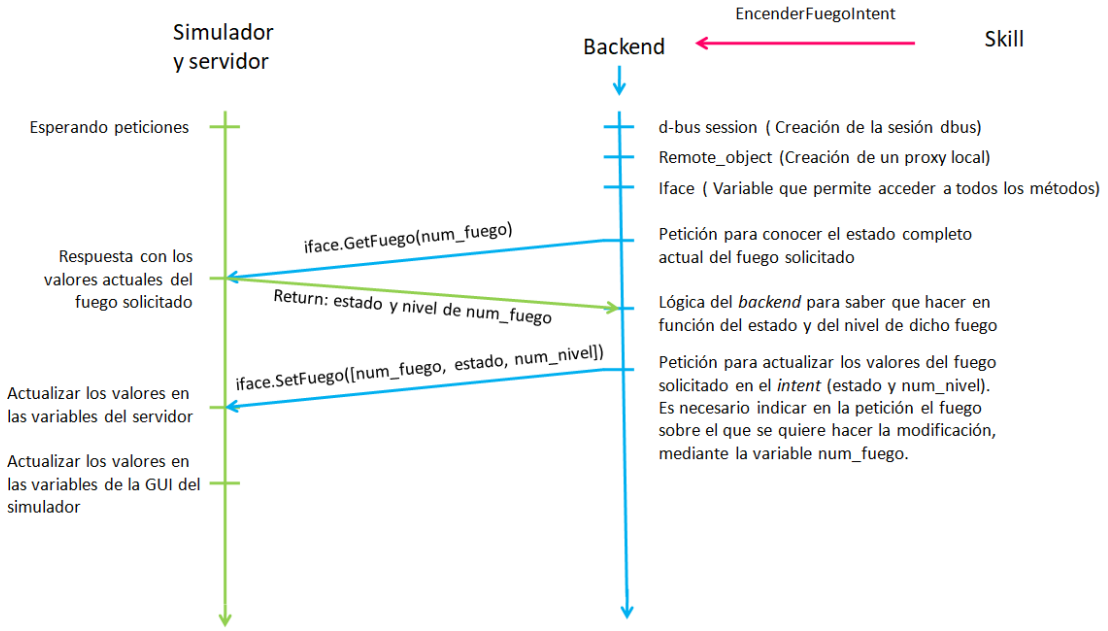

Figura 19: Diagrama de comunicación.

Para poder tener dos procesos funcionando a la vez, simulador y servidor, se ha creado el servidor como un *thread deamon* y el simulador es el *thread* principal. Un t*hread deamon* muere automáticamente cuando el *thread* principal termina, de esta manera no hay que esperar a que terminen el resto de *threads*, que en ambos casos son bucles infinitos. Al hacer que la interfaz gráfica sea el *thread* principal, cuando se cierra la ventana Tkinter del simulador, terminan ambos procesos.

3.5 Registrar la Raspberry Pi.

Es necesario registrar el dispositivo que va a hacer de 'Amazon Echo' como producto en AVS y crear un perfil de seguridad. Para hacerlo se siguen los pasos explicados en la página web de Amazon https://developer.amazon.com/es-ES/docs/alexa/alexa-voice-service/input-avs-credentials.html. Una vez completado el procedimiento, se descarga un archivo llamado *config.json* que contiene la ID del cliente, proporcionando los *tokens* de acceso que permiten interactuar con Alexa.

El dispositivo registrado, tiene acceso a todas las skills que se han desarrollado en la cuenta en la que está registrado. De esta manera puede accederse a ellas sin la necesidad de publicarlas.

3.6 Pruebas y resultados

3.6.1 Comunicación Frontend- Backend

Las primeras pruebas que se hicieron fueron entre el *backend* y el *frontend* para comprobar la comunicación, para ellas se uso la web de Amazon con la plataforma Test, que permite interactuar con la *skill* por voz o texto. Este sistema permite simular un Amazon Echo, a falta de tener un dispositivo que pueda comportarse como tal.

La comunicación se realiza con ngrok y en una de las pruebas se descubrió un problema con las URLs de Estados Unidos. Ngrok proporciona diferentes URL dependiendo de la región que se elija, tras lanzar varias veces la *skill,* una de ellas no se produjo comunicación con el *endpoint*, no se detectaba que recibiese ninguna petición POST, y, tras varios intentos reseteando la URL, se descubrió que no daba como *endpoint* válido ninguna dirección de la región de Estados Unidos; al cambiar a Europa se solucionó el problema. Además, teniendo en cuenta que la *skill* se lanza desde Zaragoza,España; es más lógico utilizar los servidores de Europa.

3.6.2 Ejecutar la skill desde la Raspberry Pi

Hasta el final del proyecto, se usó la página Test para evitar problemas de detección de la voz con el micrófono o de sonido con el altavoz. Una vez que se comprobó el funcionamiento completo de la *skill* y la comunicación, se lanzó la *skill* desde la Raspberry Pi, utilizando toda la librería de Alexa que se había instalado en la Raspberry Pi. Para poder hacer esto, la Raspberry Pi tuvo que ser registrada como producto en la Amazon Voice Service.

Este registro como producto fue el primer paso del proyecto, se hizo en febrero. Cuando se probó con el proyecto terminado, no detectaba la *wakeword* 'Alexa' cuando lo decía el usuario; sin embargo, si se ponía el comando 't'(que pone a escuchar al sistema) y el usuario hacía su petición, sí que funcionaba correctamente. Esta prueba se realizó en Junio, varios meses después y las licencias estaban obsoletas y se solucionó fácilmente haciendo una actualización de ellas.

3.6.3 Video

En el enlace disponible se puede ver una demostración del funcionamiento del sistema completo. En él, se observa cómo es la interacción entre el usuario y Alexa y las distintas peticiones que pueden hacerse; también se muestra cómo se actualiza la información en la GUI de la cocina a partir de las solicitudes del usuario, y que en todo momento se conoce el estado de la cocina, tanto si el control se hace manualmente, como por voz.

https://www.youtube.com/watch?v=sWDBgjcIgSA

4. CONCLUSIÓN Y LÍNEAS FUTURAS

4.1 Conclusión

Se ha podido realizar un sistema que dispone de una interfaz de usuario por voz para el control del simulador de la cocina de inducción, que era el objetivo del proyecto.

La interacción entre el usuario y el asistente virtual es fluida, se ha incluido un número amplio de frases que reconoce en cada tipo de petición para que sea una conversación lo más natural posible. Alexa responde cada vez que el usuario le pide algo indicando que la tarea se ha llevado a cabo con éxito, y en ocasiones dando información de la modificación que ha hecho y cuál era el estado anterior.

El diseño de la interfaz gráfica representa correctamente los controles que se pueden hacer en una cocina de inducción real, y presenta la información de forma clara y visual. Los datos se actualizan al instante, tanto si son ejecutados a través de los botones de dicha interfaz, o si las peticiones se realizan por comandos de voz. Esto es gracias a que se ha logrado la comunicación correcta entre las distintas partes del proyecto, backend - simulador y servidor a través de D-bus, controlando además el acceso de manera que no se superpongan las peticiones por voz con el control manual; y backend-frontend con ngrok, que crea un puerto seguro al que conectarse desde fuera del dispositivo.

El sistema hardware montando cumple la función de Amazon Echo. El array circular de micrófonos permite una gran capacidad de detección de voz, desde distintos ángulos y hasta una distancia de 5 metros sin necesidad de elevar la voz.

Como conclusión final se ha conseguido crear un sistema que simula como podría ser un control real de una cocina de inducción por voz, para demostrar las ventajas que tendría este tipo de interfaz. Por tanto, se puede decir que se han cumplido los objetivos iniciales del proyecto.

4.2 Futuro Proyecto

En este apartado se exponen posibles desarrollos futuros sobre el proyecto que se ha realizado. Estos desarrollos están orientados sobre todo a un sistema comercial, a partir de la base definida en este proyecto.

- Utilizar un servidor con una Base de Datos de usuarios como *backend*:
  Con la arquitectura planteada, el *backend* reside en la propia cocina que se quiere controlar, sin embargo, en un sistema comercial, esto significaría tener una *skill* diferente para cada usuario, de forma que cada una tenga como *backend* su propia cocina.

  Si se quiere hacer un sistema comercial, se necesitaría disponer de un servidor con una base de datos que almacene la información de todos los usuarios y las direcciones de las cocinas. De esta forma, se usa el servidor como *backend*, cuando algún usuario invoque la *skill*, se identifica quién es y se busca la dirección de la cocina que tiene asociada. A partir de ahí, ya se puede comunicar el *backend* con esa cocina.

Dado que se usa el sistema de reconocimiento de voz de Amazon, se podría usar también la base de datos que proporciona AWS conocida como RDS.

- Trasladar el proyecto a una cocina de inducción real
  La línea futura de este proyecto sería poder llevar a cabo un sistema real a partir de lo que se ha diseñado hasta ahora. Para poder controlar una cocina real se sustituiría en el backend las direcciones D-Bus del simulador por la de los drivers de la cocina.

- Añadir personalización y funcionalidades.
  Si se desarrolla el sistema comercial, se puede añadir más opciones que hagan más atractivo el producto de cara al consumidor. Añadir funcionalidades, como por ejemplo un temporizador que controle el tiempo que tiene que estar encendido un fuego, antes de apagarlo o de modificar el nivel. Otra opción es añadir personalización, de forma que conozca el nombre del usuario, y se dirija a él con ese nombre.

Bibliografía

[1] D. Watkins, «Report Detail». https://www.strategyanalytics.com/access-services/devices/connected-home/smart-speakers-and-screens/reports/report-detail/smart-speaker-shipment-and-installed-base-forecast-by-voice-os-by-region-2014-to-2024?slid=1095302&spg=4 (accedido jun. 24, 2020).

[2] H. L. Rufiner y D. H. Milone, «Sistema de reconocimiento automático del ha- bla», p. 28, 2004.

[3] «Sociedad de Estudios Vascos (San Sebastián, Spain) - 1996 - Ciencia, tecnología y cambio social en Euskal Herr.pdf». Accedido: jun. 24, 2020. [En línea]. Disponible en: http://gtts.ehu.es/gtts/NT/fulltext/RodriguezEtal96.pdf.

[4] «IBM Archives: IBM Shoebox», ene. 23, 2003. //www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html (accedido jun. 24, 2020).

[5] «The Rise of Voice: A Timeline», *Witlingo*. https://www.witlingo.com/the-rise-of-voice-timeline/ (accedido jun. 24, 2020).

[6] J. Pastor, «A Google se le atragantan los altavoces inteligentes: cae en picado mientras Amazon y Alexa crecen sin parar (y cuidado con Baidu)», *Xataka*, nov. 15, 2019. https://www.xataka.com/perifericos/a-google-se-le-atragantan-altavoces-inteligentes-cae-picado-alexa-crece-parar (accedido jun. 24, 2020).

[7] «Amazon Alexa Official Site: What is Alexa?» /es-ES/alexa (accedido jun. 24, 2020).

[8] «ngrok - secure introspectable tunnels to localhost». https://ngrok.com/product (accedido jun. 24, 2020).

[9] «dbus». https://www.freedesktop.org/wiki/Software/dbus/ (accedido jun. 24, 2020).

[10] «elj2005-130-D-BUS.pdf». Accedido: jun. 24, 2020. [En línea]. Disponible en: https://www.ee.ryerson.ca/~courses/coe518/LinuxJournal/elj2005-130-D-BUS.pdf.

[11] F. Lundh, «An Introduction to Tkinter», p. 324.

[12] B. Chaudhary, *Tkinter GUI Application Development HOTSHOT*. Packt Publishing Ltd, 2013.

[13] «USB Audio Streaming : UMA-8 USB mic array - V2.0». https://www.minidsp.com/products/usb-audio-interface/uma-8-microphone-array (accedido jun. 24, 2020).

[14] «Buy a Raspberry Pi 3 Model B+ – Raspberry Pi». https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ (accedido jun. 24, 2020).

[15] «FrontPage - Raspbian». https://www.raspbian.org/FrontPage (accedido jun. 24, 2020).

[16] «Raspberry Pi Downloads - Software for the Raspberry Pi», *Raspberry Pi*. https://www.raspberrypi.org/downloads/ (accedido jun. 24, 2020).

[17] M. Richardson y S. Wallace, *Getting Started with Raspberry Pi*. O'Reilly Media, Inc., 2012.

[18] «Welcome to Flask-Ask — Flask-Ask documentation». https://flask-ask.readthedocs.io/en/latest/ (accedido jun. 24, 2020).

[19] «Host a Custom Skill as a Web Service | Alexa Skills Kit». /es-ES/alexa/techdoc-template (accedido jun. 24, 2020).

[20] «ngrok - download». https://ngrok.com/download (accedido jun. 24, 2020).

[21] «Alexa Skills Kit Official Site: Build Skills for Voice». /es-ES/alexa/alexa-skills-kit (accedido jun. 24, 2020).

ANEXO I: Intents y Sample Utterances.

ApagarFuegoIntent	
1	Apagar el {num_fuego}
2	Apagar el fuego {num_fuego}
3	{num_fuego} apagar
4	Apaga {num_fuego}
5	Apaga el {num_fuego}
6	Apaga el fuego {num_fuego}

Figura 20: *Sample Utterance* del *Intent* ApagarFuegoIntent

EncenderFuegoIntent	
1	Pon el fuego {num_fuego} al {num_nivel}
2	Enciende el {num_fuego} a {num_nivel}
3	Pon el fuego {num_fuego} a {num_nivel}
4	Enciende el fuego {num_fuego} a {num_nivel}
5	Enciende el {num_fuego} al {num_nivel}
6	Pon el {num_fuego} al {num_nivel}
7	Fuego {num_fuego} al {num_nivel}
8	Enciéndelo al {num_nivel}
9	Enciende el {num_fuego}
10	Pon el fuego {num_fuego}
11	Pon el {num_fuego} al {num_nivel}
12	Ponlo al {num_nivel}
13	El fuego {num_fuego} al {num_nivel}
14	Fuego {num_fuego} al {num_nivel}
15	El {num_fuego} al { num_nivel}

Figura 21: *Sample Utterance* del *Intent* EncenderFuegoIntent

SubirFuegoIntent	
1	Subir {num_fuego} a {num_nivel}
2	Subir el {num_fuego} al {num_nivel}
3	Sube {num_fuego} a {num_nivel}
4	Sube el {num_fuego} al {num_nivel}
5	Sube el fuego {num_fuego} al {num_nivel}

Figura 22: *Sample Utterance* del *Intent* SubirFuegoIntent

BajarFuegoIntent	
1	Baja el fuego {num_fuego} a {num_nivel}
2	Baja el {num_fuego} al {num_nivel}
3	Baja {num_fuego} a {num_nivel}
4	Bajar el {num_fuego} al {num_nivel}
5	Quiero que bajes el {num_fuego} a {num_nivel}

Figura 23: *Sample Utterance* del *Intent* BajarFuegoIntent

AltoIntent	
1	Enciende el fuego {num_fuego} a fuego alto
2	Enciende el fuego {num_fuego} a alto
3	Enciende el {num_fuego} a fuego alto
4	Enciende el {num_fuego} al alto
5	Sube el {num_fuego} a fuego alto
6	Pon a fuego alto el {num_fuego}
7	Pon a alto el {num_fuego}
8	Pon alto el fuego {num_fuego}
9	El {num_fuego} a fuego alto
10	Pon el fuego {num_fuego}
11	{num_fuego} al fuego alto
12	El fuego{num_fuego} a fuego alto

Figura 24: *Sample Utterance* del *Intent* AltoIntent

BajoIntent	
1	Baja el {num_fuego} a fuego bajo
2	Baja el {num_fuego} a nivel bajo
3	Baja el fuego {num_fuego} a bajo
4	Baja el fuego {num_fuego} a fuego bajo
5	Enciende el {num_fuego} a fuego bajo
6	Enciende el fuego {num_fuego} a fuego bajo
7	Enciende el {num_fuego} a fuego bajo
8	El fuego {num_fuego} a fuego bajo
9	El fuego {num_fuego} a bajo
10	El {num_fuego} a fuego bajo
11	Pon el {num_fuego} al fuego bajo
12	Pon el fuego{num_fuego} a fuego bajo
13	Pon el fuego {num_fuego} a bajo
14	Pon el {num_fuego} a bajo

Figura 25: *Sample Utterance* del *Intent* BajoIntent

MedioIntent	
1	Pon el fuego{num_fuego} a fuego medio
2	A fuego medio el {num_fuego}
3	Pon a medio el fuego {num_fuego}
4	Enciende el fuego {num_fuego} a fuego medio
5	El {num_fuego} a fuego medio

Figura 26: *Sample Utterance* del *Intent* MedioIntent

MaxIntent	
1	{ num_nivel}
2	El { num_nivel}

Figura 27: *Sample Utterance* del *Intent* MaxIntent

SeguirSiIntent	
1	si
2	vale
3	yes

Figura 28: *Sample Utterance* del *Intent* SeguirSiIntent

SeguirNoIntent	
1	nada
2	no

Figura 29: *Sample Utterance* del *Intent* SeguirNoIntent

ApagarTodoIntent	
1	Apaga la cocina
2	Apaga todo
3	Apaga todo los fuegos
4	Apagar todos los fuegos
5	Apagar todo
6	Apagar la cocina

Figura 30: *Sample Utterance* del *Intent* ApagarTodoIntent